Safety, Security

e Resiliência

Prof. Dr. Iaçanã Ianiski Weber

Confiabilidade e Segurança de Software

98G08-4

(Material com referências abertas; verificar licenças de figuras antes de incorporar.)

Índice

Índice

Por que isso importa em Engenharia de Computação?

- Sistemas modernos são **ciberfísicos** e **conectados**: falhas + ataques podem gerar **danos físicos**.
- Dependability (confiabilidade) não é "só bug": envolve **hardware, software, rede, humano, processo**.
- **Convergência**: safety (acidental) + security (malicioso) + resiliência (sobrevivência e recuperação).

Objetivo da aula: fornecer um **framework** para identificar e justificar requisitos (safety/security) e decisões arquiteturais (resiliência).

Dependability/Trustworthiness: visão unificadora (taxonomia clássica)

Ideia-chave

Dependability é um conceito guarda-chuva que inclui atributos como: *reliability, availability, safety, integrity, maintainability* (e, ao considerar security, entra *confidentiality*).

- **Atributos (o que queremos)**: disponibilidade, confiabilidade, segurança funcional, integridade, etc.
- **Ameaças (o que pode dar errado)**: *fault* $\rightarrow$ *error* $\rightarrow$ *failure*; e ataques.
- **Meios (como alcançamos)**: prevenção, tolerância a falhas, remoção de falhas, previsão/estimativa.

Fonte-base: Avizienis et al., IEEE TDSC 2004.

Safety vs Security: definição e diferença operacional

Safety (acidental)

- Foco: **evitar dano** a pessoas/ambiente.
- Causa típica: **falha não intencional** (bug, desgaste, erro humano).
- Pergunta: "*o que pode causar um acidente?*" (hazard analysis)

Security (malicioso)

- Foco: **proteger ativos** contra adversários.
- Causa típica: **ação intencional** (exploit, engenharia social, supply chain).
- Pergunta: "*como um atacante consegue isso?*" (threat modeling)

Interseção crítica: ataques podem gerar **hazards** (ex.: sabotagem de controle).

Ameaças: falhas vs ataques (vocabulário mínimo rigoroso)

Mundo de falhas (dependability clássica)

- **Fault**: defeito/causa (HW/SW/humano).
- **Error**: estado incorreto interno.
- **Failure**: desvio observado do serviço esperado.

Mundo de ataques (security)

- **Threat**: adversário + capacidade + intenção.
- **Vulnerability**: fraqueza explorável.
- **Exploit**: método para violar uma propriedade.

Ponte entre os mundos

Ataque explora **vulnerabilidade** e induz **erro** (estado incorreto), resultando em **falha** (failure) que pode virar **acidente** (safety).

Risco: por que "Probabilidade × Impacto" é só o começo

- Em prática, risco depende de: **ameaças**, **vulnerabilidades**, **condições predisponentes**, **probabilidade de ocorrência** e **impacto**.
- Processos maduros usam: **framing** → **assessment** → **response** → **monitoring**.

Modelo mental

$$\text{Risk} = f(\text{Threats},\ \text{Vulnerabilities},\ \text{Likelihood},\ \text{Impact},\ \text{Context})$$

Referência sugerida (security risk): NIST SP 800-30 Rev.1.

Índice

Métricas clássicas: Reliability e Availability

Reliability (sobrevivência até t)

$$R(t) = P(T > t)$$

Para taxa de falhas constante λ (modelo exponencial):

$$R(t) = e^{-\lambda t}$$

Availability (tempo operacional)

$$A \approx \frac{MTBF}{MTBF + MTTR}$$

- MTBF: tempo médio entre falhas
- MTTR: tempo médio de reparo/recuperação

Sugestão de leitura: notas de curso "Dependable Systems" (HPI) para definições e intuição.

Integridade, Safety e Security: métricas e "o que medir"

- **Integrity**: correção e não corrupção (dados/estado/comando).
- **Safety**: probabilidade de **dano** + severidade (frequentemente via classes de risco).
- **Security**: não há uma única métrica; use **objetivos** (CIA/AAA), risco, cobertura de controles, e métricas operacionais (MTTD/MTTR).

Atenção

Métricas precisam ser **amarradas a um objetivo** e a um **modelo de ameaça/uso**. Sem isso viram "números bonitos".

Trade-offs típicos (e por que eles aparecem)

- **Autenticação** pode aumentar **latência** (impacto em controle em tempo real).
- **Criptografia** aumenta custo/energia; pode afetar disponibilidade em HW fraco.
- **Redundância** melhora **availability**, mas pode piorar **reliability** se introduzir mais componentes/falhas.
- **Atualizações** melhoram security, mas podem impactar safety (certificação, regressões).

Engenharia = justificar trade-offs

Não é "ter tudo", é **definir o que é aceitável** e **provar com evidência**.

Índice

Safety: conceitos operacionais (não só definição)

- **Hazard**: condição/estado do sistema que, combinada com o ambiente, pode levar a dano.
- **Accident/Loss**: dano efetivo (evento de perda).
- **Safety constraint**: restrição que deve ser mantida para evitar hazards.

Safety não é "zero risco"

É **reduzir risco a um nível aceitável** (e justificar isso).

Ciclo de vida e padrões (o que a indústria exige)

- Safety-critical usa **ciclo de vida** + **gestão de configuração** + **rastreabilidade**.
- Padrões variam por domínio: industrial (*IEC 61508*), automotivo (*ISO 26262*), aeronáutico (*DO-178C/ED-12C*).

Artefatos típicos

Hazard log, safety requirements, arquitetura de mitigação, V&V plan, evidências, **safety case**.

Caixa de ferramentas de Hazard Analysis (comparativo)

Técnica	**Uso típico**	**Quando brilha**
FMEA / FMEDA	falhas por componente e efeitos	HW/SW com boas fronteiras
FTA	combinações lógicas levando a evento topo	justificar mitigação e redundância
ETA	consequências pós-evento	barreiras e escalonamento
HAZOP	desvios de intenção (guidewords)	processos/indústria
STPA	controle/feedback em sistemas complexos	software-intensivo/ciberfísico

STPA é central em cursos modernos de System Safety (ex.: MIT OCW).

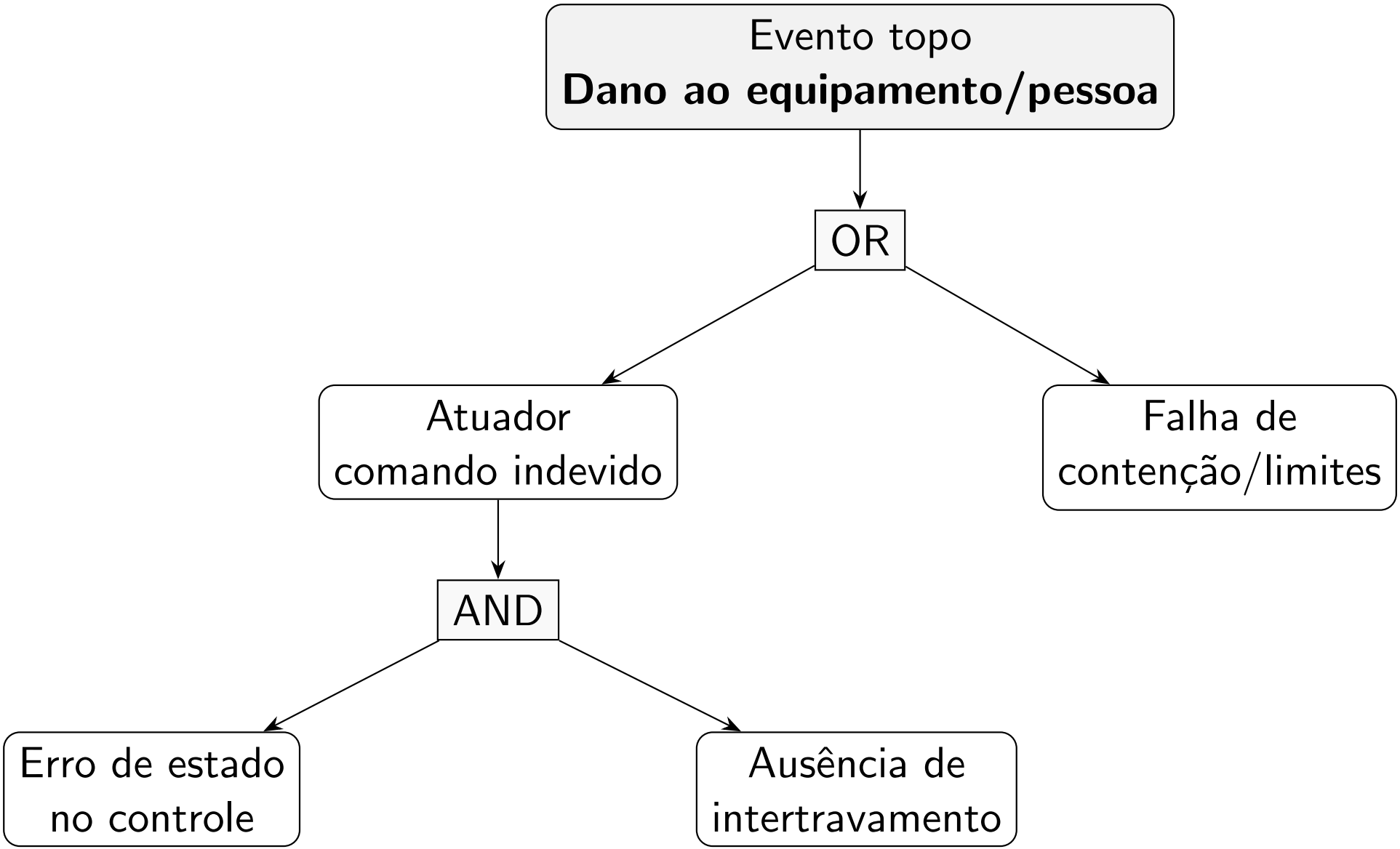
Exemplo de FMEA (template mínimo)

Item	Modo de falha	Efeito	Mitigação
Sensor	saturação/leitura travada	controle recebe valor incorreto	plausibility check, redundância
Firmware	overflow em cálculo	comando fora de faixa	saturação + testes + revisão
Comunicação	perda/atraso	comando stale	timeout + fail-safe

Ponto crítico

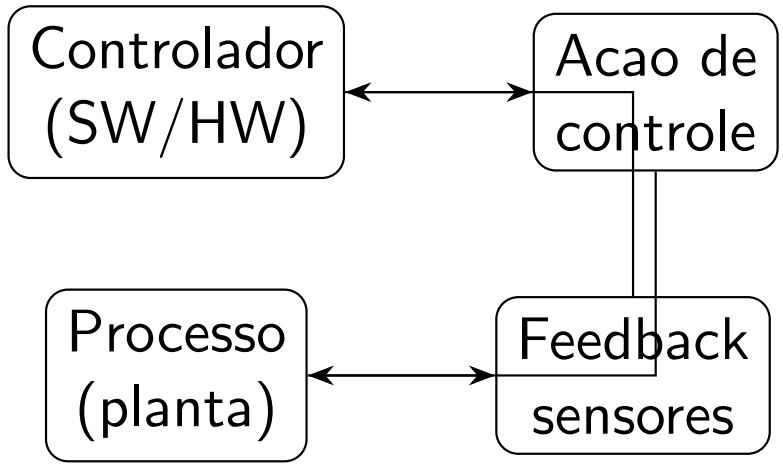
Mitigação vira **requisito verificável** (e entra no plano de V&V).

Fault Tree Analysis (FTA): do acidente às causas



STPA (STAMP): quando o sistema e controle + feedback

- Acidentes nao sao so cadeia de falhas; sao violacoes de restricoes em sistemas de controle.
- STPA identifica acoes de controle inseguras e cenarios causais.
- Causas tipicas: feedback inadequado, modelo mental incorreto, latencias e atrasos.



Resumo pratico

Foco em restricoes de controle e em como validalas por projeto, monitoramento e operacao.

Índice

Security: propriedades e fronteiras de confiança

- Objetivos clássicos: **CIA** (Confidentiality, Integrity, Availability) + **AAA** (AuthN, AuthZ, Accounting).
- Em embarcados/ciberfísicos, **integridade de comando** e **tempo** (freshness) são críticos.
- Segurança começa com **trust boundaries**: onde dados/comandos cruzam domínios.

Threat Modeling (visão além do STRIDE "tabela")

Artefatos úteis

- Diagrama de fluxo de dados (DFD) + fronteiras de confiança
- STRIDE como **heurística** por componente/fluxo
- **Attack Trees** para raciocínio "objetivo do atacante → caminhos"
- Misuse/abuse cases (requisitos negativos)

Boa prática

Threat modeling é **iterativo**: atualiza com mudanças de arquitetura, incidentes e testes.

Risk Assessment em security: processo (NIST SP 800-30)

- Defina: escopo, ativos, missão, tolerância a risco (*risk framing*).
- Identifique: **threat sources/events**, **vulnerabilities**, **predisposing conditions**.
- Estime: **likelihood** e **impact**.
- Determine: **risk** e priorize respostas.

Entrega típica

Registro de riscos + plano de tratamento (mitigar, transferir, aceitar, evitar) + monitoramento.

Referência sugerida: NIST SP 800-30 Rev.1.

Operacionalizando: NIST Cybersecurity Framework (CSF) 2.0

- CSF 2.0 organiza práticas em 6 funções: **Govern, Identify, Protect, Detect, Respond, Recover**.
- Útil para mapear requisitos/controles e responsabilidades organizacionais.

Uso em disciplina

Transformar ameaças/vulnerabilidades em **controles** e em **planos** (detecção, resposta, recuperação).

CSF 2.0 (inclui versão em português): NIST, 2024.

Mecanismos típicos (embedded/IoT): o "mínimo profissional"

- **Secure boot** + cadeia de confiança (root of trust).
- **Atualização segura** (assinatura, anti-rollback, A/B, recovery).
- **Proteção de chaves** (HSM/secure element/TPM quando aplicável).
- **Hardening**: superfície mínima, privilégios mínimos, isolamento (MPU/MMU).
- **Observabilidade**: logs auditáveis, telemetria, detecção de anomalia.

Verificação em security: do requisito ao teste

- Requisitos (ex.): “toda atualização deve ser autenticada e íntegra”.
- Evidências:
  - revisão de design + threat model atualizado
  - **testes** (fuzzing de parser, testes de downgrade/rollback)
  - **análise** (SAST/DAST, revisão de dependências, SBOM)
  - **pentest** orientado ao modelo de ameaça

Índice

Quando security vira safety (e vice-versa)

- Ataque $\Rightarrow$ violação de integridade/tempo $\Rightarrow$ estado perigoso (**hazard**).
- Mitigações de safety podem criar riscos de security (ex.: portas de manutenção, bypass).
- Mitigações de security podem piorar safety (ex.: DoS por autenticação mal projetada).

Mensagem

Projetar separadamente é arriscado: o correto é **co-analisar** interações e trade-offs.

STPA-Sec / co-análise: um caminho moderno

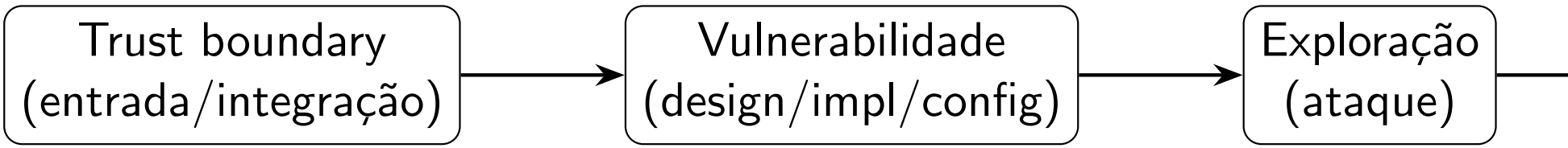
- Extensões de STPA aplicam o raciocínio de **controle/feedback** também para **ameaças cibernéticas**.
- Benefício: evidencia como vulnerabilidades impactam restrições de safety.

Uso prático

Definir **constraints** (safety) e **constraints** (security) no mesmo modelo, derivando requisitos integrados.

Sugestões (fontes abertas): trabalhos e materiais do ecossistema MIT/Leveson sobre STPA e security.

Fluxo de incidente (com safety + security): versão mais completa



Pergunta de engenharia: em quais setas você coloca controles?
(prevent/detect/respond/recover)

Assurance case: como justificar "está seguro o suficiente"?

- Ideia: **argumento estruturado** + **evidências** + **premissas/escopo**.
- Em safety: safety case é comum; em security: security case é mais difícil (adversário muda).
- Em sistemas conectados: ideal é **assurance contínua** (monitorar + atualizar evidência).

Índice

Resiliência: definição operacional

Definição (cyber resiliency)

Capacidade de **antecipar**, **resistir**, **recuperar** e **adaptar** a condições adversas, estresses, ataques ou compromissos envolvendo recursos cibernéticos.

- Resiliência não substitui safety/security: ela **complementa** quando a prevenção falha.
- Entra como requisitos de: **degradação graciosa**, **fail-operational**, **reconfiguração**, **continuidade**.

Referência sugerida: NIST SP 800-160 v2r1 (Developing Cyber-Resilient Systems).

Padrões arquiteturais de resiliência (mapa prático)

Antecipar/Resistir

- segmentação/zonas
- least privilege / isolamento
- diversidade (N-version, heterogeneidade)
- rate limiting, circuit breakers
- validação robusta (parsers)

Recuperar/Adaptar

- rollback (A/B), recovery mode
- checkpoints, reinicialização controlada
- degradação graciosa (safe state)
- reconfiguração/roteamento alternativo
- observabilidade + aprendizado pós-incidente

Métricas de resiliência (operacionais e úteis)

- **MTTD** (Mean Time To Detect) e **MTTR** (Mean Time To Recover/Repair).
- **RTO/RPO** (tempo/quantidade de perda aceitável) — mais comum em serviços.
- **Taxa de sucesso de rollback**, **cobertura de telemetria**, **tempo em modo degradado**.

Ponto crucial

Sem instrumentação (logs/telemetria), não há resiliência: você não detecta nem recupera.

Resiliência como "engenharia ao longo do ciclo de vida"

- Resiliência exige processos: governança, gestão de riscos, resposta a incidentes, melhorias contínuas.
- Em organizações maduras, security/resiliência são tratadas como **práticas contínuas** (não "projeto fechado").

Sugestão de leitura: relatórios do CMU/SEI sobre práticas de segurança/resiliência ao longo do ciclo de vida.

Índice

Caso base: controlador embarcado (contexto e fronteiras)

- MCU + firmware; sensores/atuadores; barramento CAN/Ethernet; gateway; backend OTA.
- **Hazards**:
  - comando fora de faixa (atuador crítico)
  - operação fora do envelope (ex.: tempo/temperatura)
- **Threats**:
  - injeção de mensagens no barramento
  - comprometer pipeline de atualização (supply chain / servidor)

Do requisito ao mecanismo: exemplo de cadeia de confiança (OTA)

Requisito (security + safety)

Atualização deve ser **autenticada** e **atômica**, com **rollback seguro** e sem permitir downgrades.

- Assinatura digital da imagem + verificação no bootloader
- Esquema A/B (slot ativo + slot candidato)
- Anti-rollback (monotonic counter / versão mínima)
- Health check pós-boot; se falhar $\rightarrow$ rollback automático

Exercício de engenharia (em grupo): hazard log + threat model

1. Liste **3 hazards** e classifique severidade (qual o dano?).
2. Liste **3 threats** e descreva pré-condições do atacante.
3. Para cada hazard, proponha **1 safety constraint**.
4. Para cada threat, proponha **1 controle** (prevent/detect/respond/recover).

Exercício avançado: FTA + Attack Tree para o mesmo evento

Evento topo (comum)

Comando indevido no atuador crítico (causando hazard).

- Construa uma **FTA** (falhas acidentais) chegando a causas básicas.
- Construa uma **Attack Tree** (ação intencional) chegando a caminhos de ataque.
- Compare: **quais mitigações servem para ambos?** quais são específicas?

Índice

Resumo (com densidade)

- Use uma **taxonomia** para organizar atributos, ameaças e meios (dependability).
- Safety: de hazard analysis $\rightarrow$ constraints $\rightarrow$ requisitos $\rightarrow$ evidência.
- Security: threat modeling + risk assessment + controles + verificação contínua.
- Resiliência: projetar para **falhar com segurança** e **recuperar** (antecipar, resistir, recuperar, adaptar).
- Engenheiros justificam decisões via **trade-offs** e **evidência**.

Gancho: próxima aula — Teste de software (V&V, níveis e planejamento)

Conexão direta

Tudo que discutimos hoje precisa virar **evidência**: e V&V é o principal mecanismo para gerar evidência técnica.

- **Verification**: "construímos certo?" (conforme especificação/requisitos).
- **Validation**: "construímos a coisa certa?" (necessidade/uso real).
- Níveis: unitário, integração, sistema, aceitação; e testes não-funcionais (segurança, robustez, desempenho).
- Planejamento: rastreabilidade requisito $\leftrightarrow$ teste, critérios de saída, cobertura e evidências.

Referências abertas (para aprofundar) I

- Avizienis, Laprie, Randell, Landwehr. *Basic Concepts and Taxonomy of Dependable and Secure Computing*. IEEE TDSC, 2004.
- NIST SP 800-30 Rev.1. *Guide for Conducting Risk Assessments*, 2012.
- NIST CSF 2.0 (inclui versão PT-BR), 2024.
- NIST SP 800-160 v2r1. *Developing Cyber-Resilient Systems*, 2021.
- MIT OpenCourseWare (Leveson). *System Safety* (STPA e materiais correlatos).
- University of Cambridge. *Security Engineering* (notas de curso abertas).
- HPI (Hasso Plattner Institute). *Dependable Systems* (definições e métricas).
- CMU/SEI. Relatórios sobre *security/resilience* ao longo do ciclo de vida.